DES-1: AN INTER-ACTIVE CONTINUOUS SYSTEM SIMULATION LANGUAGE

by

Michael Alex Kalashian

United States
Naval Postgraduate School

THESIS

DES-1: AN INTER-ACTIVE CONTINUOUS SYSTEM SIMULATION LANGUAGE

by

Michael Alex Kalashian

June 1969

DES-1: An Inter-active Continuous System Simulation Language

by

Michael Alex Kalashian
Second Lieutenant, United States Marine Corps
B.S., United States Naval Academy, 1968

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1969

ABSTRACT

There are strong tutorial advantages to Digital Computer Simulation of Control System's problems. This is particularly true where such simulations do not require sophisticated programming techniques and where the user may directly interact with his problem. The purpose of this study was to develop such a capability for the Naval Postgraduate School's direct-access Computer System.

The installation was to be accomplished using the DES-1 Simulation Language and an SDS 9300 Digital Computer. The DES-1 software requires a special DES-1 Console for optimum performance. Due to the lack of this Console, a reformulation of the language was necessary. This process involved simulating the Console and revising the language to operate with existing hardware.

The language was re-written and the revised system has been installed as an operating system. Complete documentation is available in the DES-1 Programming Manual, which was prepared as part of this study.

TABLE OF CONTENTS

I. INTRODUCTION

Those who use Analog Computers for the purpose of solving scientific and engineering problems often find the need for more precise results. Others have found that the precision offered by the Analog Computer was sufficient, but that the problems encountered in program construction were enormous. Another group expressed the desire for a method of checking their Analog Computer results. A solution to these problems is the Digital Simulator. With this simulator a check solution, with greater accuracy and simplicity of programming, is possible. A Digital Simulator is a Digital Computer equipped with a Simulation Language.

Users of the Analog Computer at the Electrical Engineering Computer Laboratory, at the Naval Postgraduate School, were among the people expressing a need for solutions to the above problems. An answer to this need became available with the acquisition of a Simulation Language. The language, DES-1 (Differential Equation Solver), could be used in conjunction with the SDS 9300 Digital Computer available in the laboratory. This Simulator combination has been used successfully in many installations.

In order to utilize this combination, measures were necessary to satisfy DES-1 hardware requirements. Because of limited equipment availability, it was decided to modify these requirements. This was accomplished by re-writing the DES-1 language.

In a discussion of a technical nature it is necessary to define the subject of inquiry. Chapter II introduces the concept of Simulation Languages. A description of the evolution of Simulation Languages

is contained in Chapter III. Chapters IV through IX relate the procedures followed and the results of modifying the available version of DES-1.

II. INTRODUCTION TO SIMULATION LANGUAGES

Simulation Language may seem a rather remote, undefined, or misunderstood term to a person encountering it for the first time. Simulation is the key term. To simulate is to represent something by means of something else, such as representing the set of all numbers by all positive integers, or a physical system in terms of a mathematical model. This last example is the main object of Simulation as considered here.

A. ANALOG COMPUTERS

For many years the Analog Computer has been used to simulate engineering systems. This computer uses voltages to simulate physical variables. The voltages are generated by inter-connection of electrical components. These components include resistors, capacitors, and amplifiers. The components are connected so that the voltages in the computer obey certain mathematical relationships. These relationships are generally differential equations. These equations form the mathematical model of the physical system. The number of simultaneous equations that can be simulated in parallel is limited only by the number of components available on the Computer. Because of this capability, "Differential Analyzer" was the name originally given to today's Analong Computer.

Without going into explicit details of analog operation, it will suffice to note that there were and still are many shortcomings to

analog computation. Among these are scaling and precision. These inequities are caused by the continuous nature of the computer. The simulated physical variable, voltage, is directly dependent on the nature of the computer components. Therefore, a problem solution is subject to inaccuracies caused by the tolerances in computer components.

B. DIGITAL COMPUTERS

Digital Computers have been in existence for many years, but only limited effort was applied to using a sequential machine to solve continuous and, in general, time-dependent problems. The Digital Computer, in its basic form, is not well suited to the job of simulating a continuous system. It remained for a few astute programmers and engineers to alter this basic form. With this thought and its subsequent implementation, Digital Simulation was conceived but, as will be seen, not born.

C. DIGITAL SIMULATION

From the system-design standpoint, the ability to eliminate or reduce the major problems in a design before production is a necessity. It is very unfeasible to build something in order to discern its workability. Initially this problem was solved using a mathematical model of the proposed system. This simulation made testing and evaluation more practical; but it was still no easy task. The Analog Computer eased this task with its ability to simulate a mathematical model. This leads to Digital Simulation, which is simulation of an Analog Computer by a Digital Computer. It must be noted that many of the initial probes into the field of Digital Simulation were

not intended to simulate Analog Computers. They were to aid an analog programmer in initial setup of his simulation. Such problems as finding scaling factors and establishing initial values of problem variables were greatly reduced. Digital solutions also provided a rough means of checking analog results. With the idea of Digital Simulation formulated in very raw form it remains to be seen what has evolved, since R. G. Selfridge's start in 1955.[1]

D. SIMULATION CRITERIA

Before going into specific programs and languages that have appeared, some criteria for effective simulation will be examined. These criteria are rather general and at times contradictory, but as in any design situation, a trade-off approach is necessary. As the first criterion, Digital Simulation must be accurate: that was one of the main reasons for wanting something better than the Analog Computer. Simplicity is also important. An analog programmer should be able to use the language without becoming an expert digital programmer. The system trade-off involves the requirement for flexibility. The Simulation Language should have the capability of handling many different types of problems, but this flexibility requirement implies complexity, which should be avoided.

One major advantage of the Analog Computer that should be retained is the ease of interaction between the man and his problem. This "hands-on" approach is extremely valuable to a programmer who would like to make variations in the problem parameters while the problem

---

[1]Linebarger, R. N. and Brennan, R. D., "A Survey of Digital Simulation," Simulation, v. 3, p. 25, December 1965.

is in progress, or between individual runs. As any user of a widely used Digital Computer knows, computer time is very valuable. Also, the machinery is very complex. This restricts, in many cases, a "hands-on" availability. These criteria are general and minimal, as stated, but the particulars of each will become apparent with the explanation of the specific programs.

III. SIMULATION LANGUAGES

The term Digital Simulation has been used to describe two different simulation processes. The first, although insignificant for this study, is the simulation of systems which are in themselves discrete. An example of this type is a data processing system such as SIMSCRIPT. The second is the simulation of continuous systems. The scope of this study is limited to the latter.

A. TYPES OF DIGITAL SIMULATION

Within Digital Simulation, henceforth restricted to simulation of continuous systems, there are two different approaches. The first and most widely used is the so-called block-oriented language. This refers to the simulation of the actual inter-connection of components, or blocks on an Analog Computer. The other approach is the procedural type language. This type system essentially solves differential equations in equation form. A combination of the versatility of both these types is the pursuit of modern simulation designers. This approach is referred to as the Continuous System approach. An extremely large problem in the development of these languages was that most were developed by individuals, for individual use. Even at this, the

situation would have been better if these inovators had communicated their results to others. Hence, as shall be seen, the evolvement of Digital Simulation is a history of unnecessarily repeated labor and lost lessons learned by mistakes.

B. SIGNIFICANT SIMULATION LANGUAGES

At this time a few of the significant Simulation Languages will be examined. These languages have led the field to its present state. First to be discussed will be the block-oriented languages, which were also first historically.

1. Selfridge

As mentioned, the first published work on simulation was offered by R. G. Selfridge in 1955. His system did not use very sophisticated methods or machinery. This use was limited by state-of-the-art hardware. A major shortcoming was the use of fixed-point arithmetic. This made scaling necessary at times, but still greatly increased the range and accuracy of results. This program established many of the basic ideas of Digital Simulation.[2]

2. DEPI (Differential Equation Pseudo code Interpreter)

This program appeared in 1957. It was an expansion of Selfridge's original unnamed program. Of major interest was the use of a much more sophisticated integration scheme, the fourth-order Runge-Kutta method. Selfridge used Simpson's Rule. Also improved were some of the block-operators of the original program. The first

---

[2]Clancey, J. J. and Fineberg, M. S., "Digital Simulation Languages: A Critique and a Guide," AFIPS Conference Proceedings, 1965 Fall Joint Computer Conference, Part 1, v. 27, p. 23, 1965.

program allowed amplifier inputs to be pre-multiplied by constants only. DEPI allowed multiplication by variables.[3]

3. ASTRAL (Analog Schematic Translation to Algebraic Language)

This program, appearing in 1958, was designed to be directly programmable from an analog diagram. This included amplifier sign reversals, fixed input gains, etc. This convention allowed the maximum amount of simplicity to the analog programmer. This program was the first to use floating-point arithmetic. This completely eliminated scaling and also greatly increased the precision of results. Another inovation was that this system had its own compiler. It generated FORTRAN statements which were executed via the FORTRAN compiler. As a direct consequence of this, FORTRAN arithmetic statements were allowable. In principle this program was a forerunner of the Continuous System Languages.[4]

4. DYSAC (Digitally Simulated Analog Computer)

This program, released in 1961, was an improved version of DEPI-4, which was an improved version of DEPI. The only difference was the addition of floating-point arithmetic, as in ASTRAL. DYSAC also improved the formatability of its predecessor. It allowed mnemonic characters, etc. Also of interest was its concern for the user. It adopted and used many diagnostic indicators. These were a big aid in program de-bugging.[5]

5. MIDAS (Modified Integrated Digital Analog Simulator)

This program, introduced in 1964, was the most significant to date. Its importance was derived from being not only a good program,

---

[4]Linebarger and Brennan, p. 27.

[5]Linebarger and Brennan, p. 29.

but also it was the first widely recognized and used Simulation Language. This program was, by its own definition, a sophisticated version of DAS, which had been instituted a year earlier. DAS used blocks with mnemonic codes, as in DYSAC. It also used its own compiler, as in ASTRAL.

MIDAS was very similar to DAS, but used a more sophisticated integration method: a fifth-order predictor-corrector with variable step size. It also provided for the reduction of algebraic loops, i.e., equations of the form

$$X = e^{-Xt} + K.$$

Its predecessors had to avoid them.[6]

Up to this time no languages had been implemented that satisfied the man-machine criteria established in Chapter II. In 1964 two programs of significance were introduced that directly attacked this problem.

6. Pactolus

This slightly updated version of MIDAS was offered by IBM. This update included direct user control from the computer console. This system was not extremely powerful in the computation sense, but it did give the programmer a means to interact with the machine.[7]

7. DES-1 (Differential Equation Solver)

This system, the main object of this Thesis, was introduced by the Scientific Data Systems Corporation (SDS). It included its

---

[6]Harnett, R. T., Sansom, F. J., and Warshawsky, L. M., "MIDAS," Simulation, v. 3, p. 17, September 1964.

[7]Brennan, R. D. and Sano, H., "Pactolus," AFIPS Conference Proceedings, 1964 Fall Joint Computer Conference, v. 26, p. 299, 1964.

own console and the high computation capability of the SDS 9300 Computer. This program will be discussed in detail in Chapter IV.

Although the line separating block-oriented and procedural type languages is very vague, the following are considered examples of the latter type. This division is based mainly on the allowable format of program statements. The block-oriented languages are built around operators and have statements of the form

3 SUM X, Y, Z

where 3 is the block number, SUM is the operator, X is the output, and Y and Z are the inputs. The procedural type is built around an expression type language, such as FORTRAN. This type has statements of the form

X = Y + Z.

The difference is obviously minimal.

8. DSL/90 (Continuous System Modeling Program)

The most notable of these expression type programs was intended for use on the IBM 7090-class computers. It is now available on the IBM 360 series also. This system is based on expression type notation, as in FORTRAN, with unrestricted format. The system was modeled on the capabilities of MIDAS. The program achieves its flexibility because of its use of a FORTRAN compiler, as in ASTRAL.[8]

9. CSSL (Continuous System Simulation Language)

The Continuous System approach has evolved into a combination of both the expression-based and block-oriented based systems. This merge was the work of the Simulation Councils, Inc. which, through

---

[8]Linebarger, R. N., "DSL/90," Simulation, v. 7, p. 108, September 1966.

its monthly publication Simulation, joined the two camps of thought to produce a prototype language: CSSL. This language incorporated all the good points of past languages, except one important one. It did not provide for any man-machine interface. Its main strong point was the retention of program simplicity, while maximizing Digital computation methods. This language was not actually implemented. It was intended, rather, as a guideline to future designers.[9]

10. SL/1

A result of the CSSL language was a simulation language from SDS for use in its SIGMA 5/7 computers. This language uses all the capabilities of FORTRAN along with a system for implementing Analog Computer functions. This system allows Hybrid operation, which is the inter-connection of analog and digital devices for simultaneous dependent solutions. This capability enhances SL/1's already powerful repertoire.

C. SUMMARY

As has been shown, Digital Simulation has made significant advances since its inception in 1955. To an engineer with an available simulation system, complete dependence on an Analog Computer for solution of continuous type problems is greatly reduced. As with anything new, people are over-cautious and it may take some time before the value of Digital Simulation is fully realized.

---

[9]The SCi Simulation Software Committee, "The SCi Continuous System Simulation Language," Simulation, v. 9, p. 281-303, December 1967.

IV. INTRODUCTION TO THE DES-1

The DES-1 in its operating form is a high-speed general-purpose Digital Computer hardware-software complex, with a specially oriented set of "operators" that allow it to perform in a manner similar to an Analog Computer. This system is built around a DES-1 software package, an SDS 9300 Computer, and a special operator console.

A. CAPABILITIES

This package completely satisfies the criteria stated in Chapter II. These criteria required simulation to be more accurate and as easy to use as an Analog Computer. Retention of some of the desirable features of the Analog Computer, such as the man-machine interface, was also desired. Perhaps the greatest capability of the DES-1 over other simulation systems is its man-machine interface. This interface allows the user to be "on-top" of his problem at all times. The advantages of this type of operation are manifold. Reduction of problem "turn-around" time and the capability to monitor problem progress are the most important. These features greatly reduce time expended to obtain a problem solution. Another large advantage of this system is the capability for Real Time and Hybrid operation. This allows a programmer to force a Digital Computer to operate in time synchronization with an Analog Computer. This greatly expands the allowable scope of system problems.

B. DESIGN CONCEPT

The DES-1 system was designed around the concept of operators. An operator is a pre-programmed set of instructions that perform tasks

similar to those accomplished by the Analog Computer, such as Dead Band, Delay, Integration, and Function Generation. The DES-1 also includes an array of utility subroutines such as Sine, Cosine, and Square Root.

The DES-1 provides the capability for use of almost any existing input/output device including the Line Printer, Typewriter, Brush Recorder, Oscilloscope, etc. The DES-1 Real Time capability allows a user to perform Hybrid operations (simultaneous programming of a problem on suitably interfaced Analog and Digital Computers). The DES-1 also provides for inclusion of any special user subroutines, which may be retained within the system software for application by any subsequent user.

C. COMPARISON TO THE STATE-OF-THE-ART

As with any system, the DES-1 has inherent shortcomings. Being a digital device it is limited in frequency response. Also, any DES-1 user must compete with regular digital programmers for computer time. This time is usually less available on a Digital Computer than on an Analog Computer. Although the DES-1 system was released in 1955, conceptually it lacks very little compared to the state-of-the-art today. As mentioned in Chapter III, the newest Digital Simulators have capabilities for both block-oriented and procedural type statements. The DES-1 is a block-oriented type program but it does incorporate some expression type notation, such as arithmetic equations. However, its man-machine interface capability is not duplicated in the newer languages. The CSSL language is admittedly more sophisticated, but it offers little more than a greater freedom of format in input programs.

V. CONSTRUCTION OF THE DES-1 SOFTWARE PACKAGE

The DES-1 software package was designed to be installed in an SDS 9300 Digital Computer with a minimum of 8,192 (8K) words of memory and floating-point arithmetic. The software package was designed to be independent of any other software packages. It does not require any extra processors or peripheral programs. The system is made up of eight parts that with the SDS 9300 Computer, the DES-1 Console, and available input/output equipment comprise a DES-1 computer system.

The eight parts that make up the software package are listed and briefly described below.

A. SYSGEN

The first part is the DES-1 System Generation (SYSGEN). This program loads the remainder of the software package in order to generate a system file on a Rapid Access Drum (RAD). It also loads the Master Executive part of the DES-1, which resides in core memory at all times and controls subsequent access to the RAD.[10]

B. MASTER EXECUTIVE

The next part is the above-mentioned Master Executive program. This program not only contains utility routines used in all phases of DES-1 operation, but also loads all the remaining parts for use during appropriate phases of operation. There are three phases of operation in the DES-1 system: the Compile phase, the Load phase, and the Run

---

[10] See Figure 1.

phase. According to operator selected modes, the remaining parts of DES-1 are loaded when required.[11]

C. COMPILER

The DES-1 Compiler, another part of the system, is loaded during the Compile phase of operation. It compiles the user's source program and produces an object program that the computer can execute. The Compiler is also loaded when the Update feature of DES-1 is used.

D. UPDATE

The fourth part of the system is the Update program. This program is called by the Master Executive whenever the operator initiates an Update control code. The Update will load the updated corrections and transfer control to the Master Executive, which will call the Compiler to re-compile the new source program. After the source program has been compiled into an object program, the Master Executive initiates the next phase of operation, the Load phase.

E. SUBLOD

The Load phase of operation is handled by the Subroutine Loader (SUBLOD), which is called by the Master Executive. This phase includes loading of the object program and all necessary subroutines into memory. At the completion of this phase the Master Executive loads the Run Time Executive, thereby initiating the third phase of operation, the Run phase.

---

[11] See Figure 2.

F. RUN-TIME EXECUTIVE

The sixth part of the DES-1 package is the Run-Time Executive. This program controls the actual execution of the user's program. This program contains the routines to execute the analog type modes of operation. These modes are used in the following manner. The RESET mode is used to establish problem initial conditions as required. When initial conditions are satisfied, the READY mode is entered; this indicates that the problem is ready for solution. At this time, the OPERATE mode may be entered which will cause the execution of the problem. The HOLD mode may be entered at any time for inspection of problem progress or parameters. This part of the program also contains routines to execute mode changes and problem variable input/output control codes.[12]

G. SUBROUTINE LIBRARY

The seventh part contains the DES-1 Library Subroutines, such as integration schemes, Sin/Cos, etc. The eighth part is the DES-1 Hybrid Library. This part contains the necessary subroutines to allow Hybrid operation in conjunction with a suitably interfaced Analog Computer.

The DES-1 system allows for the inclusion of user-written subroutines. When this option is exercised, these user subroutines become the ninth part of the DES-1 system. When included, they are handled similarly to the two DES-1 subroutine packages. All subroutines are loaded into memory as needed during the Load phase of operation, by the Subroutine Loader.

---

[12] See Figure 3.

A general Flow Diagram description of the DES-1 system is presented in Figure 4. For a more detailed description see the DES-1 Programming Manual, which was written in conjunction with this Thesis.

VI. NATURE OF THE PROBLEM

It was desired to make the DES-1 Digital Simulator available as an operating system at the Electrical Engineering Computer Laboratory, at the Naval Postgraduate School. The above-mentioned Computer Laboratory has available an SDS 9300 Digital Computer and a version of the DES-1 software package that had been adapted by SDS for use at another installation.

As has been stated, one of the main advantages of the DES-1 system is its man-machine interface. This is accomplished via the special DES-1 Console. A complete description of this console is included in Chapter VIII. It is to be noted that the system can be used without this special console, but so doing reduces its effectiveness to that of an ordinary Digital Computer solving differential equations by numerical methods.

The initial problem faced was the non-availability of this DES-1 console. The importance of having the capabilities afforded by the console was recognized and it was decided to simulate the console as much as possible. The simulation of the console was limited to hardware available and to realizable changes in the DES-1 software package.

The original DES-1 system was designed to be used in a facility that had at least three magnetic tape units available. The version available to the laboratory at the Naval Postgraduate School had been altered to operate from two magnetic tape units and a Rapid Access

Drum (RAD). This hardware requirement was met by the laboratory, but some alterations became necessary, as will be detailed in Chapter VIII.

The original system used the DES-1 Console as the medium for achieving its wide variety of output methods. With the console, that was equipped with Digital-to-Analog converters, outputs were possible on a multi-channel Recorder, Oscilloscope, X-Y Plotter, and Digital Display. It was desired to retain this output capability as much as possible.

It was of prime importance that the system retain its proven capabilities to the fullest extent. At the same time, however, it was desired to retain the simplicity of operation necessary to attract potential users, namely the Analog Computer programmers.

VII. PROCEDURES FOLLOWED FOR SYSTEM INSTALLATION

The previously mentioned version of the DES-1, available at the Naval Postgraduate School, was in a form unsuitable for use by the Computer Laboratory.

A. PROGRAMMING LANGUAGES

A brief explanation of computer programming languages is included here to enable a better understanding of the problem faced. In general, there are two basic types of programming languages. The terms higher-level and lower-level are usually employed to define the difference. A higher-level language is usually a problem-oriented language such as FORTRAN, ALGOL, or PL/1. A lower-level language is a machine or machine-oriented language. Digital Computers execute

machine language instructions that are assembled by translation or compilation of a higher-level language. Machine language is the lowest-level language. A machine-oriented or assembly language is a language that directly parallels a machine language, but it allows the use of symbolic operation codes and symbolic addresses. A machine language statement for addition, for example, might have the form

075 46000 ,

where 075 means add and 46000 is the memory location of the variable to be added. A machine-oriented language statement to achieve the same operation might have the form

ADD X ,

where X describes memory location 46000. Machine and assembly languages are usually written for particular computers or computer types. Thus, for example, IBM has a language for its computers, whereas SDS has a different language for its computers.[13]

B. DES-1 LANGUAGE

DES-1 is in itself a higher-level problem-oriented language. The DES-1 software package is written in SDS's machine-oriented language META-SYMBOL. The version available was in the standard SDS Binary, or Encoded Card Deck form. From this card deck a binary output could be obtained that would then be usable to generate the DES-1 system.

---

[13]International Business Machine Corporation Report TR 00.1663, Concepts and Terminology for Programmers, by R. W. Engels, 2 October 1967.

C. PROCEDURES

The version available required the use of SDS's Universal Binary Loader, a program that loads a binary program written on either cards or paper tape. The use of this loader restricted the DES-1 system availability to either cards or paper tape. In order to avoid as many large changes as possible and to facilitate the installation it was decided to retain this loader.

Due to the large size of the DES-1 system (some 30,000 META-SYMBOL statements) it was decided to use paper tape as the system medium. The complete system is available on two reels of paper tape. The paper tape is much faster and more manageable than cards. Both tapes can be loaded in approximately ten minutes. The thousands of cards that would be required would have taken a much greater amount of time using the available card reader.

It is anticipated that a new loader will be written to allow the DES-1 system to be available on magnetic tape, which is faster and more convenient. The paper tape system can easily be transferred to magnetic tape using existing programming techniques.

VIII. MODIFICATIONS TO OPERATING SYSTEM

A. DES-1 CONSOLE

In order to be able to use the DES-1 system as desired, certain software modifications were necessary. The basic problem involved changing the system to operate with a simulated console. It was desired to retain as many capabilities and functions performed by the console as possible. These capabilities and functions included the following:

1. A Digital Display that can display numbers as four decimal digits plus sign, and a two-digit decimal exponent plus sign;

2. Display-Select Switches that are programmable to allow digital display of problem variables;

3. Eight Sense Switches that allow the programmer to alter program execution via their settings;

4. Four Manual Potentiometers that provide for manual changing of problem conditions during problem execution;

5. A Frame-Time Selector that allows a user to specify the value of the time increment to be used in integration equations and in the delay operator;

6. A Multiple Selector that allows the user to set the time increment desired for use with the RATE2 option. This option allows different equations within a program to be solved using different time intervals;

7. A Card-Input Selector that allows control code entry via the typewriter or card reader;

8. Arithmetic and Frame-Time Overload Hold buttons and alarms;

9. Mode Selector Switches that allow a user to select or change the modes of operation; and

10. Digital-to-Analog Converters that convert digital results to a form suitable for use on continuous-type output devices.

A limited amount of hardware was available at the Computer Laboratory to use for simulation purposes; therefore most of the retained functions were programmed to use the available typewriter.

The eliminated functions were the Digital Display, the Display Selector, the Sense Switches, and the Card-Input Selector. The remaining functions and capabilities were simulated in the following manner. The Manual Potentiometers, the Frame-Time Selector, the Multiple Selector, and the Arithmetic and Frame-Time Overflow alarms were programmed to operate from the typewriter. The Digital-to-Analog conversion was programmed to use the Digital-to-Analog converters available in the laboratory. The Mode Selector switches and the Arithmetic and Frame Time Overflow Hold buttons were programmed to use the six sense switches available on the SDS 9300 console.

The original DES-1 system used hardwired input/output device instructions to achieve information transfer between the computer and the console. It was necessary, therefore, to replace these instructions with subroutines that would accomplish this transfer of information.

B. MODIFICATIONS

The following is a detailed description of the modifications made to the DES-1 program. It is noted that the modifications are not necessarily described in the order in which they were completed. The system is described in the order that it is constructed.

1. System Generation

As previously noted, the function of this part of the system is to generate a system file on the Rapid Access Drum (RAD).[14]

A problem arose only after most of the changes to be explained in ensuing sections were accomplished. The RAD system file had been set up to conform to the original size of the program. However, it was necessary to replace functions previously performed by hardware with subroutines as large as fifty instructions. These subroutines caused parts of the program to overflow their alloted RAD storage space. This problem was solved by re-designing the RAD layout for the system file. This part of the program was also re-written to accept input from two reels of paper tape. It had been designed to accept input on cards only.

2. Master Executive

The major modifications necessary to this part of the program were the changing of console-dependent operations. Also necessary were certain space changes made necessary by the change in the system's size.

Included in this part were routines to handle the following console functions:

1. The Card-Input Switch;
2. The SET-UP-Mode indicator; and
3. The Arithmetic-Overflow Hold and alarm.

The card Input Switch was altered to restrict control code input to the typewriter only. The SET-UP mode light routine was

---

[14]See Figure 1.

eliminated and replaced with a typewriter output that types, "MODE-SETUP." If an arithmetic overflow occurs, the program will check the Arithmetic-Overflow Hold switch and take appropriate measures. These measures are described in the DES-1 Programming Manual. The alarm was changed from the original light and buzzer to a typewriter output which will type, "ARITHMETIC OVERFLOW HAS OCCURRED." The sound of the typewriter is sufficient to alert an operator to this condition. The methods used for both typewriter outputs are similar.[15]

Other modifications to this part of the program include changing of the RAD-File-Description-Table (FDT) indicators. These indicators point to the proper place on the RAD when the Master Executive accesses the RAD. Also necessary was the creation of additional space to be used for instruction storage. These stored instructions are used by the Compiler and are explained in that Section. It was necessary to include them in this portion of the program because they were to be used during both the Compile and Run phases of operation. The Master Executive is the only part of the system in memory at both times. The necessary space was taken from unused memory at the top of the program.

Included in this part were subroutines for clearing the interrupts caused by the inclusion of special Hybrid operations to be explained in the Compiler section. Minor changes were also necessary in memory allocation of linkage variables. Figure 2 is a general diagram of operation within this part of the system.

---

[15] See Figure 5.

3. Run-Time Executive

As in the Master Executive the major modifications in this part were the modification of console-dependent functions. Within this part of the system, routines were included to handle the following console functions:

1. The Mode interrupts and indicators;
2. The Real-Time switch;
3. The Frame-Time Overflow Hold and alarm; and
4. The Frame-Time and Multiple selectors.

With reference to the console-simulation decisions, the following changes were made.

In the original system pushing one of the mode buttons caused a hardware interrupt that was serviced by a routine that checked to see which mode switch had been set. This routine selected the appropriate routine to enact the mode change.

In order to simulate this operation, the SDS 9300 Console sense switches were used in conjunction with a hardware interrupt (INTR 33) available on the same console. In order to execute a mode interrupt the appropriate sense switch is set and then the INTR 33 button is pushed. When the operation is initiated, the computer will go to memory location 33 and execute the instruction contained in it. This instruction was programmed to be identical to the instruction that would have been in the memory cell used by the DES-1 Console interrupt. The checking routine mentioned above was changed to check the console sense switches. INTR 33 does not need to be cleared after execution. Whenever a mode is selected the typewriter will type out a mode indicator of the form, "MODE-RESET." This typeout replaces the original mode switch backlight used on the DES-1 Console.

The Real-Time switch was replaced with a routine that initiates a typewriter request for a decision. The typewriter will type the following message, "REAL TIME:." After the operator pushes INTR 33, the computer will accept a Y for yes or an N for no, and then continue operation. This change was accomplished by inserting a routine to type the request, wait for the answer, and then take the answer and proceed accordingly. The routine replaces the Real-Time switch and operates in the following manner. When initiated by DES-1 system control, the request message is typed out. Cell 33 is then loaded with an instruction to direct the program to a routine for receiving the operator's answer. The program will then Halt, awaiting the answer. When INTR 33 is pressed, the program is directed to a routine that reloads cell 33 with the mode-interrupt instruction and alerts the typewriter for input. Reloading cell 33 is necessary because the Real-Time routine destroys the previously explained mode-interrupt director. When the operator types in his decision the program will decode the input and proceed in the proper direction.

The Frame-Time Overflow Hold and alarm were changed to operate the same as the Arithmetic-Overflow Hold and alarm.

The original DES-1 Console had digital switches that were used for the Frame-Time (DELTA) and Multiple (MULT) inputs. The Frame-Time switch was a six-digit Binary-Coded-Decimal (BCD) switch. The Multiple switch was a two-digit BCD switch. These switches were read by the Executive program and the variables DELTA and MULT set to the respective values. Binary-Coded-Decimal (BCD) characters are described by six binary digits (bits). Decimal number BCD characters have only four significant bits, therefore they require only four

bits to completely describe them. The DES-1 Console truncated each number set at the console to a string of four-bit BCD numbers which were then read by the Executive program.

In order to simulate this on the typewriter it was necessary to replace the original read instruction with a routine that would read six-bit BCD numbers from the typewriter and convert them to four-bit BCD numbers. This was accomplished in the following manner for both the Frame-Time and Multiple inputs. A routine was included, similar to that used for Real-Time that would initiate the request, "DELTA IS:" or "MULT IS:" and then receive the information via the typewriter. The conversion was accomplished by a series of register shifts and loops that truncated each digit to the required four bits before storing it. A diagram for this operation is included in Figure 5.

A small change was included in the potentiometer processor routine in conjunction with changes made in the Compiler that will be explained in that section. Changes were also made in memory allocation of many linkage variables because of the relocation of the overall system.

4. Update

The Update portion of this program is used to accept operator-initiated source program corrections. There were no changes to this part of the system.

5. Compiler

The DES-1 Compiler is used to translate the user's source program, written in the DES-1 language, into a machine-language object program. Two major changes were necessary in this program.

These changes involved the potentiometer input processor and the graphic output device processor.

A brief explanation of how the DES-1 Compiler works is included to facilitate understanding of the changes made. When compiling a source program, the Compiler produces a set of machine-language instructions for each DES-1 statement it processes. These machine-language instructions are stored on magnetic tape until the Load phase of operation when they are read into a pre-assigned area of memory. Because of the space alterations mentioned earlier, the portion of memory used for this object program was reassigned. This reassignment is what necessitated the relocation of all the linkage locations mentioned previously.

The potentiometer processor compiled a set of machine instructions that when executed during the Run phase of operation would have caused the following action:

1. The manual potentiometers were read from the DES-1 Console in the afore-mentioned four-bit BCD form;

2. The execution branched to a routine in the Run-Time Executive which processed the input for use in the problem; and

3. The execution returned to the object program.

This action was repeated for each Frame-Time-incremented value of time. This action allowed changing of the potentiometer values at will during problem operation. Unfortunately this feature had to be eliminated because of the lack of readily available hardware to implement the manual potentiometers. It was foreseen that at a later date the potentiometers on the laboratory's CI-5000 Analog Computer may be used.

It was decided to allow input of potentiometer values during the Compile phase of operation. This capability, coupled with the user's option to change the value of problem variables using the TI control code, retained the original system flexibility with only a small increase in execution time. This change was accomplished by using a routine similar to that used for Frame-Time, Multiple, and Real-Time inputs. This routine will initiate, via the typewriter, the request, "INPUT POTS." When received, the information from the typewriter, in a format specified in the DES-1 Programming Manual, will be converted to the required four-bit BCD code and stored as a machine-language instruction. When the object program is executed, the Run-Time Executive will treat the instruction in which the potentiometer value is stored as the input from the original console. This was done by changing two instructions in the Run-Time Executive, as referred to in that section. This procedure is diagrammed in Figure 6.

The original Compiler contained an output processor for each available graphic output device. The processors for the Digital Display, multi-channel Recorder Oscilloscope, and X-Y Plotter generated instructions that caused output to these devices via the converters in the DES-1 Console. As has been mentioned earlier, it was decided to retain as many of these outputs as possible. The Digital-to-Analog Converters available in the Computer Laboratory were used to accomplish the desired output.

Using this procedure it is possible to present information on any of the available Analog Computer graphic output devices. These devices include a Digital Voltmeter, a multi-channel Recorder, an

Oscilloscope, and the X-Y Plotter. Output by this method is programmed by a standard DES-1 output statement, as explained in the DES-1 Programming Manual.

To accomplish this output a new processor was written and included in place of the four original processors. This new processor generates a series of machine-language instructions. These instructions will convert a selected problem variable to the form necessary for output via a Digital-to-Analog converter. The variable will be output to an operator-selectable trunk line on the CI-5000. There are eight different outputs available on eight different trunk lines. The operator may use the outputs on any device available. The processor uses the stored instructions referred to in the Master-Executive section. This greatly reduced the size of the processor by allowing each of the eight different output statements to be processed by a single processor. A detailed description of this processor is included in Figure 7.

6. Subroutine Loader

The Subroutine Loader accomplishes loading of the object program and necessary subroutines during the Load phase of operation. The only modifications necessary in this part of the system were changes made to the Rapid-Access-Drum (RAD) File-Description-Table (FDT) indicators.

7. DES-1 Library and Hybrid Subroutines

The main problem with the subroutines was one of a mechanical nature. At some time before the initation of this project, some of the subroutine Encoded decks had been mixed up and some cards had been misplaced. After finding and correcting the mix up, on y two

subroutines remained missing. One was a Library subroutine and the other a Hybrid subroutine. The Library subroutine was reproduced using an old DES-1 listing available. The Hybrid Library contains two copies of each subroutine for use with the multiple-rate feature of the DES-1. The missing Hybrid subroutine was constructed using its nearly identical copy.

Another problem was found during testing. The Runge-Kutta-Gill (RKG) integration subroutine was incorrectly updating the independent problem variable TIME each cycle. This was corrected to conform to specifications in the DES-1 Programming Manual.

8. Program Recovery

The nature of Encoded Decks allows immediate recovery of the original system cards. All corrections were made on colored cards so that the distinction is easily accomplished. Removal of all correction cards leaves the original Encoded Decks.

IX. CONCLUSIONS

The DES-1 system available for use at the Electrical Engineering Computer Laboratory, at the Naval Postgraduate School, offers almost all the capabilities of the originally designed system. Included among the available capabilities are:

1. The system offers seven different integration schemes, from the relatively simple Trapezoidal method to the ultra-sophisticated Adams-Moulton Checking method;

2. The integration inputs have no number limit and the inputs may be sums and products of terms;

3. The system allows function generation of one, two, or three variables;

4. The system has the capability of using input/output devices of either the tabulated or graphic type;

5. The system affords a user a hands-on capability for monitoring of problem progress;

6. The operator language allows a user to program directly from block diagrams or differential equations;

7. The programming rules are simple and error diagnostics are ample to minimize time spent in problem set-up;

8. The DES-1 offers the accuracy, reliability, and versatility of a high-speed general-purpose Digital Computer; and

9. The DES-1 system offers a complete Hybrid Library for user control of Hybrid hardware.

Complete details of the DES-1 system are available in the DES-1 Programming Manual, which was prepared as part of this study.

The main capability not offered in the modified system is the capability to operate in the Real-Time mode. This mode of operation is directly dependent upon the availability of an accurate Real-Time Clock. The clock was available in the DES-1 Console in the original system. The only clock available at the time of the system modification was a sixty-Hertz clock in the SDS 9300. This was not sufficient for use in this application. The DES-1 Console was equipped with a ten-Kilohertz clock that generated pulses every one hundred microseconds. It is recommended that hardware may be altered to offer a faster clock that can serve the DES-1 system.

A slight increase in problem time was an unavoidable side effect of not using the DES-1 Console. The only really noticeable increase is associated with the potentiometer input, although it was noted that most users of the CI-5000 Analog Computer follow a similar procedure for changing potentiometers.

Because of the nature of a digital machine and the use of a Digital-to-Analog converter as the means of achieving graphical output, the graphical results are not completely continuous. The outputs do present an accurate visual result, that when correlated with tabulated results provides an extremely meaningful solution. It was found that using the problem variable TIME as one of the outputs greatly helped this correlation.

Operating procedures are detailed in the DES-1 Programming Manual. A few of the more important points are presented to illustrate the ease of operating the DES-1 system.

1. The system can be generated in ten minutes with a minimal knowledge of Digital Computers required;

2. All system equipment during problem solution, except the line printer, can be arranged to be operated by the user from a chair in front of the typewriter;

3. Operating instructions are straightforward and easy to understand; and

4. Provisions are included for memory dumps for detailed trouble shooting or to store a problem for future use.

The DES-1 was designed and has been modified with special consideration towards the user. In order for it to be an effective system it must and does satisfy criteria established for meaningful Digital Simulation. The only readily apparent advantages of the newer versions of Simulation Languages, such as CSSL and SL/1, are the more powerful Digital Computers they are designed to operate with, and the simultaneous access available to another higher-level language such as FORTRAN. This is not possible with DES-1 because of the size limitations of the SDS 9300 Computer memory. The newer systems are designed to operate on larger computers that have enough memory to include a higher-level language Compiler in addition to the

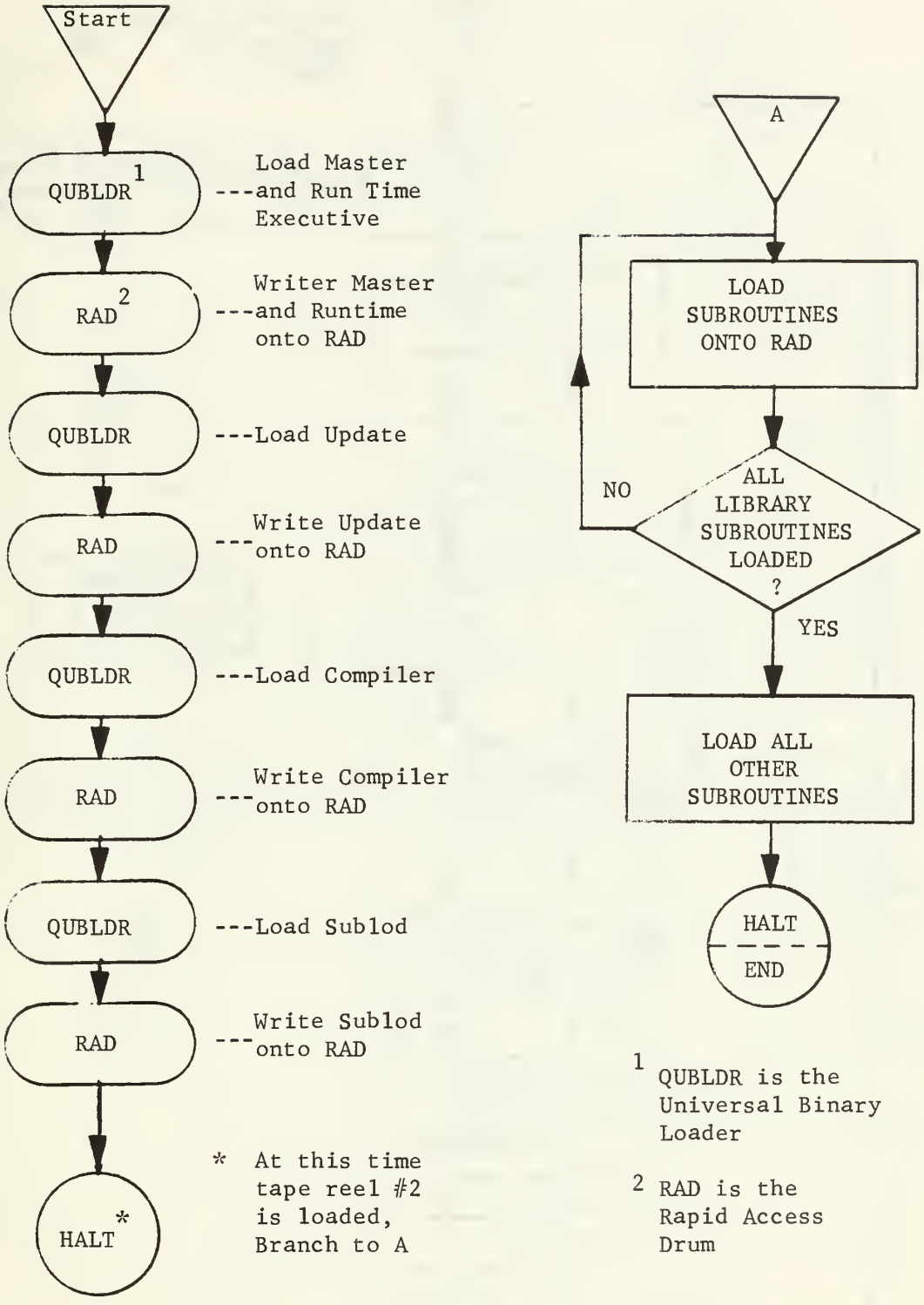
Simulation Language package. There is no denying that in certain situations these advantages would prove extremely useful and perhaps necessary. It is noted that the DES-1 system under discussion is intended for use in a basically student-oriented environment. Its capabilities over the newer languages, namely the man-machine interface, combined with its computational, etc. capabilities offer more to its intended environment than could one of the more powerful languages.

APPENDIX A

This Appendix contains the following DES-1 system Flow Charts:

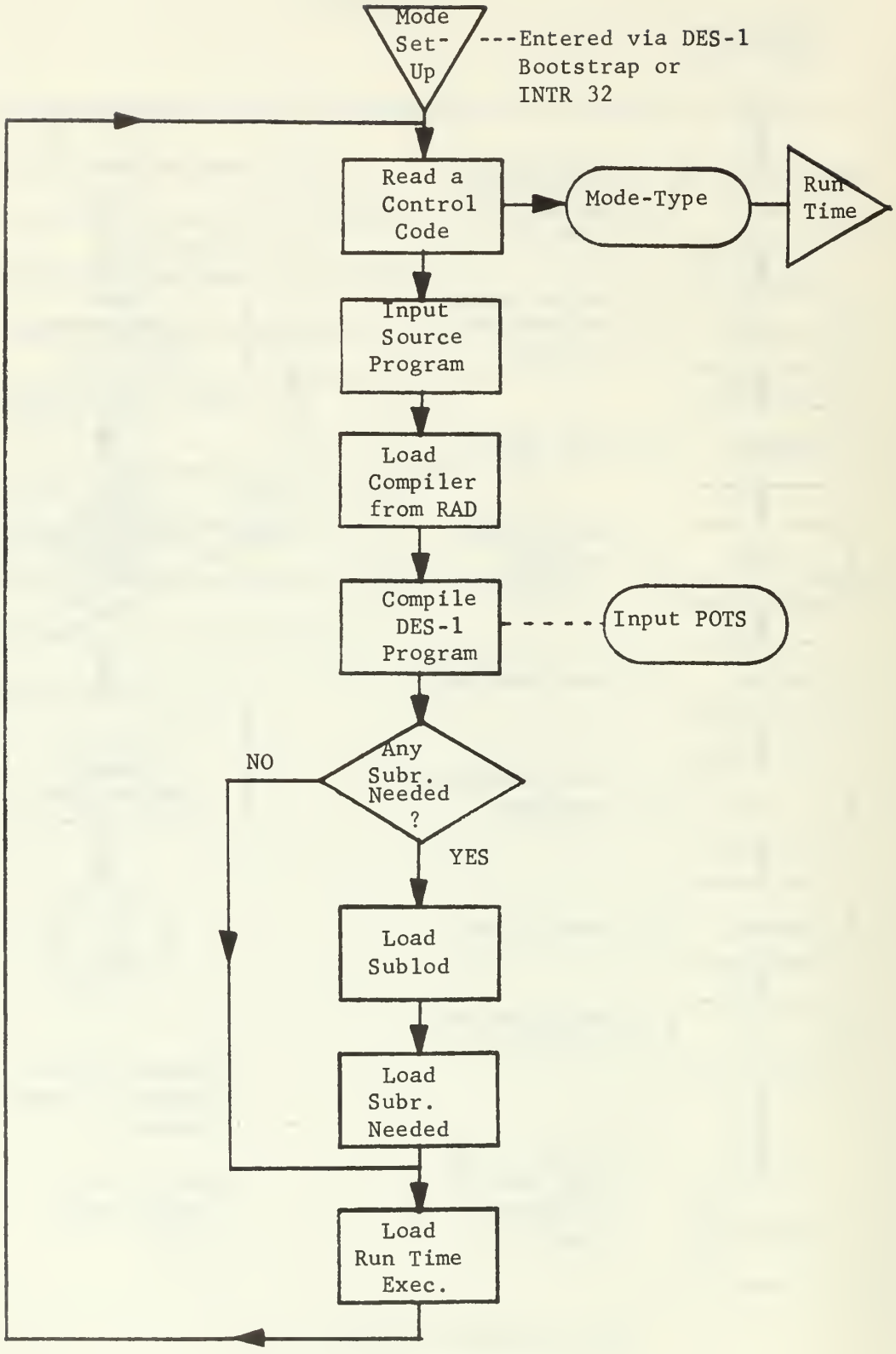
1. DES-1 System Generation;
2. The Master Executive Program;
3. The Run Time Executive Program;
4. DES-1 System Flow Diagram;
5. Typewriter Output and DELTA and MULT converters;
6. The Potentiometer Input Processor; and
7. The Graphical Output Process.

These Flow Charts are labeled Figures 1 through 7, and are referred to within the text of this Thesis.



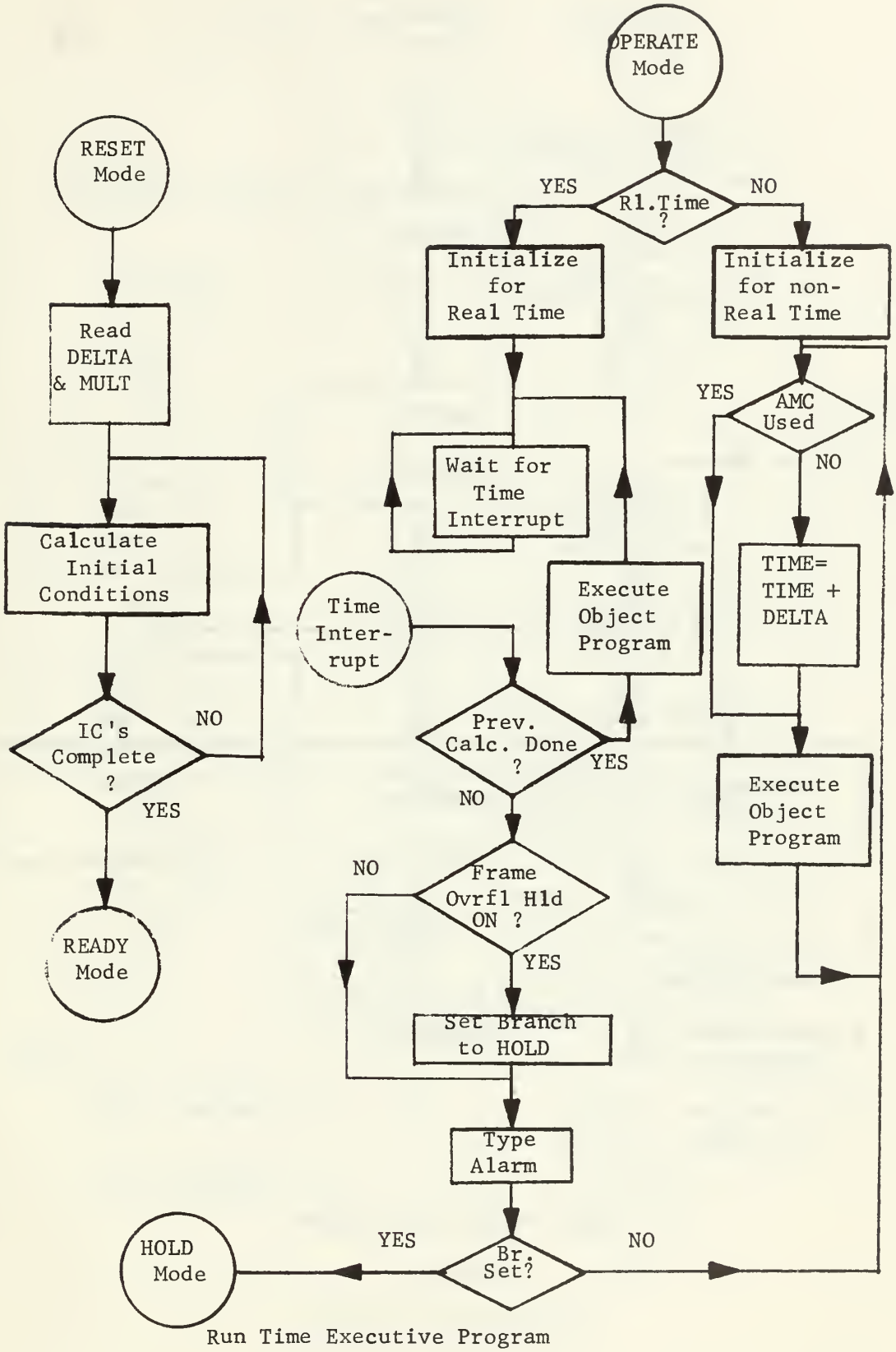
DES-1 System Generation

Figure 1



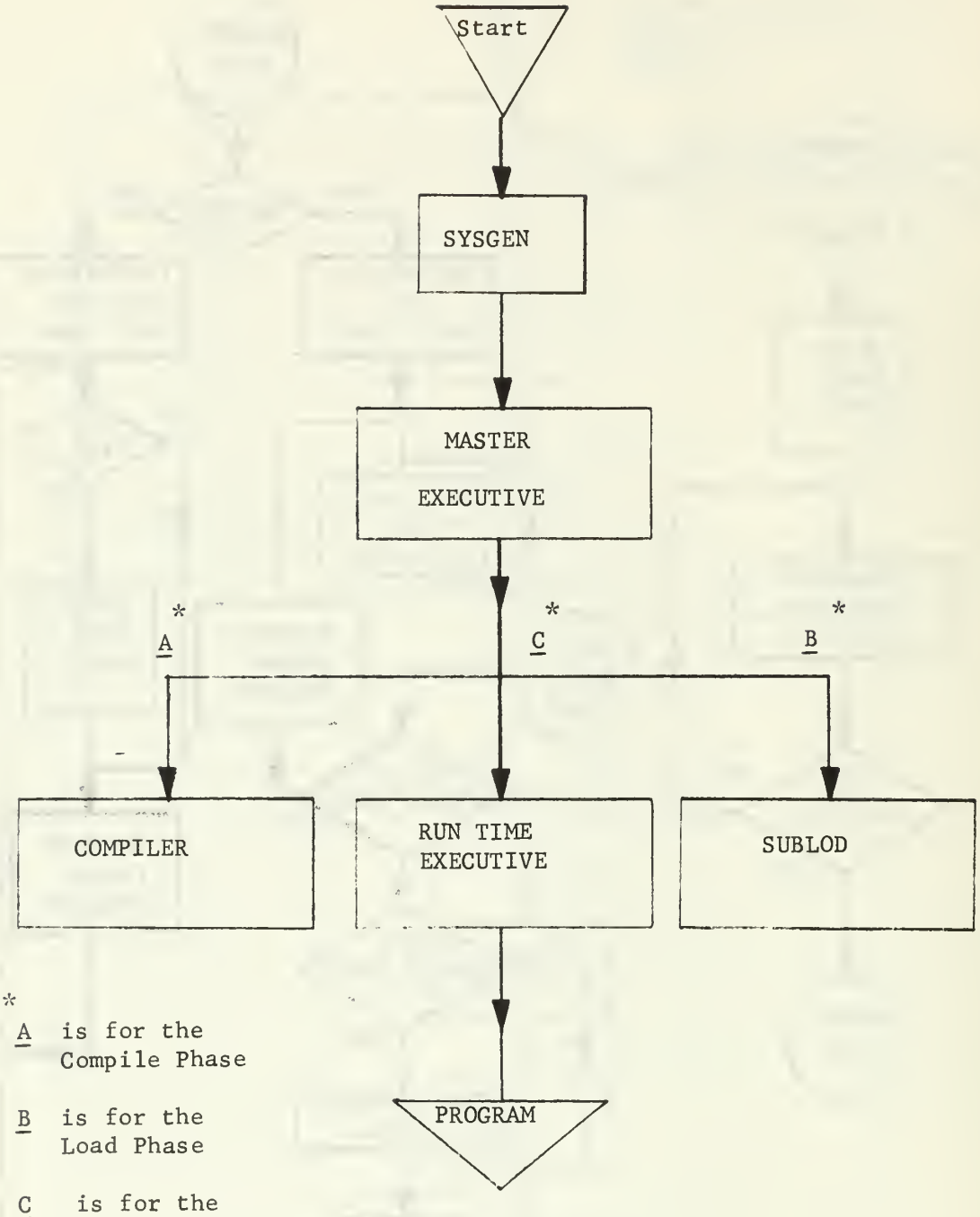
Master Executive Program

Figure 2



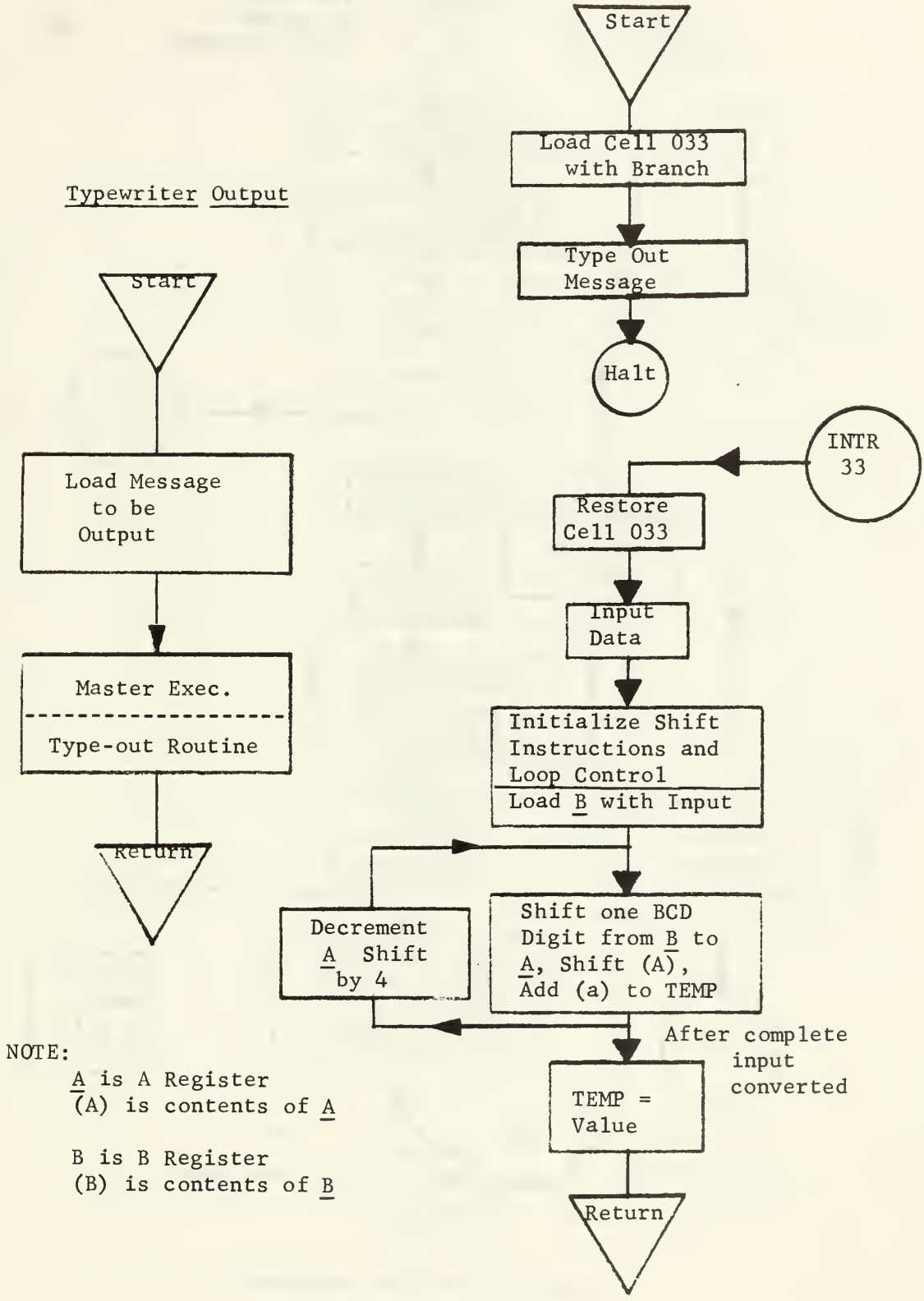
Run Time Executive Program

Figure 3



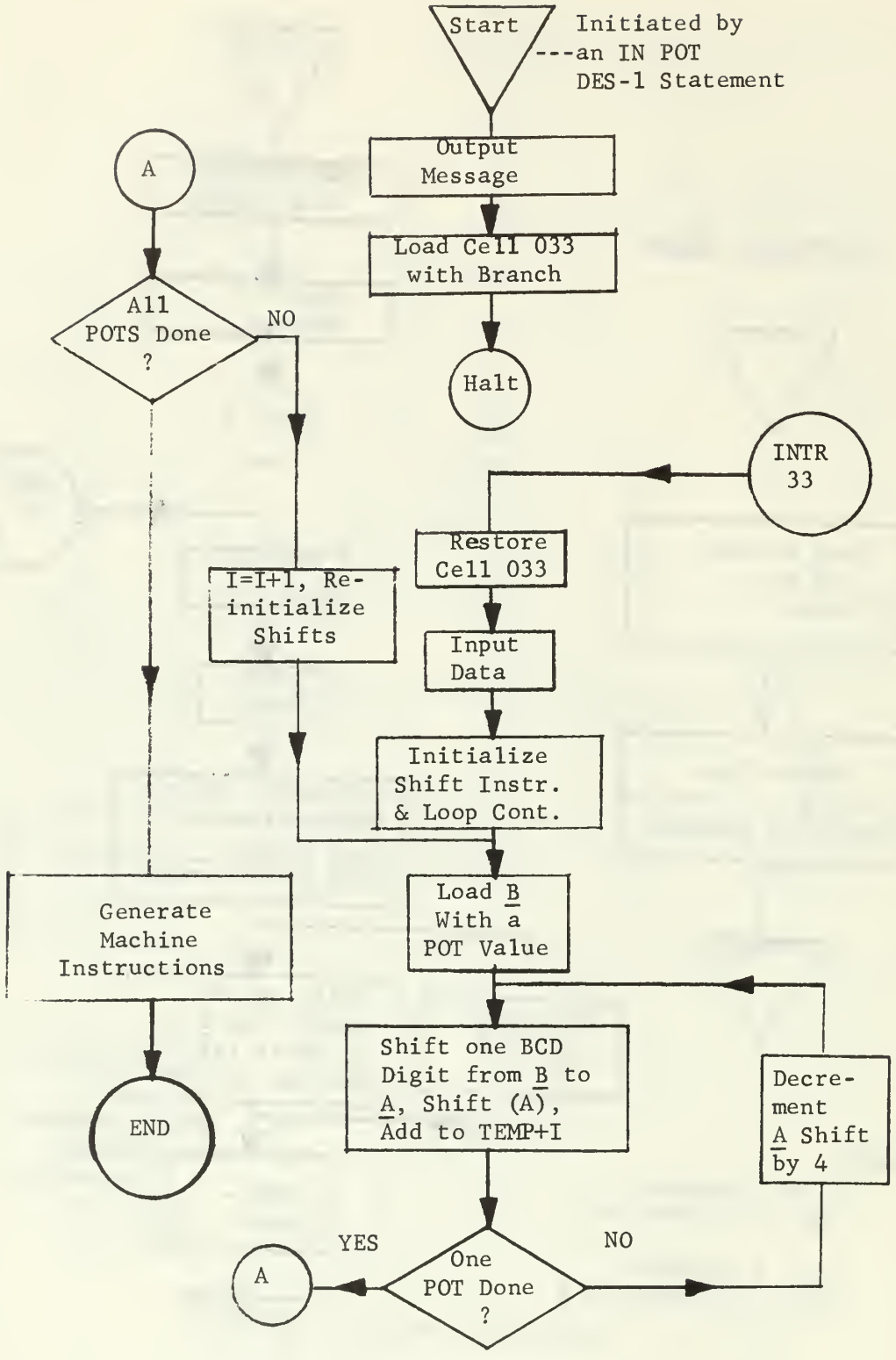
DES-1 System Flow Diagram

Figure 4



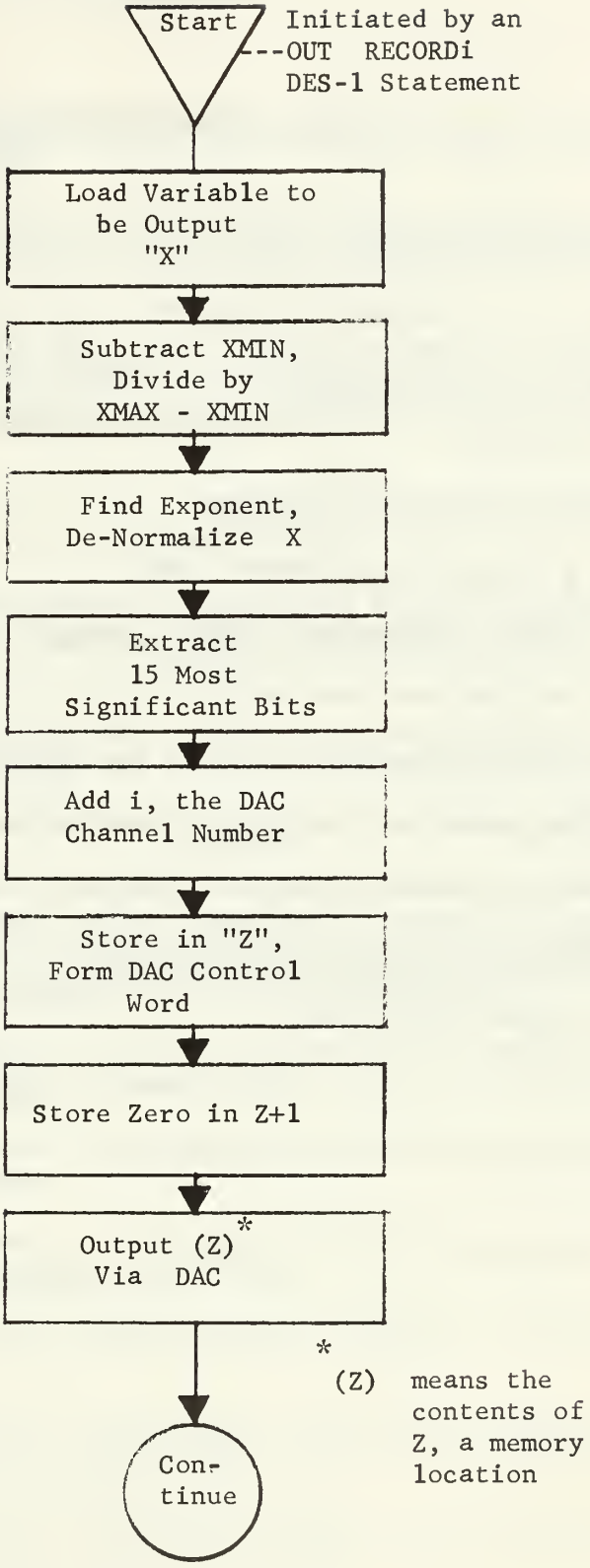
Typewriter Output and
DELTA and MULT Converters

Figure 5



Potentiometer Input Processor

Figure 6



Graphical Output Process

Figure 7

APPENDIX B

THE DES-1 PROGRAMMING MANUAL

The following is a brief introduction to the DES-1 Programming Manual. This manual was prepared by the author of this paper. The manual is a combination of the following DES-1 publications, as revised by this author:

1. DES-1 System Generation,
2. DES-1 Reference Manual, and
3. DES-1 Programming Technical Manual.

These publications were modified to reflect the changes made to the DES-1 system for installation at the Naval Postgraduate School.

This manual was compiled and copies were made in order to provide an easy and accurate reference for DES-1 users. These copies are available at the Electrical Engineering Computer Laboratory at the Naval Postgraduate School.

LIST OF REFERENCES

1. Brennan, R. D., Continuous System Modeling Programs, Pamphlet, North-Holland, 1968.

2. Brennan, R. D. and Sano, H., "Pactolus," AFIPS Conference Proceedings, 1964 Fall Joint Computer Conference, v. 26, p. 299, 1964.

3. Clancey, J. J. and Fineberg, M. S., "Digital Simulation Languages: A Critique and a Guide," AFIPS Conference Proceedings, 1965 Fall Joint Computer Conference, Part 1, v. 27, pp. 23-36, 1965.

4. Harnett, R. T., Sansom, F. J., and Warshawsky, L. M., "MIDAS," Simulation, v. 3, p. 17, September 1964.

5. International Business Machine Corporation Report TR 00.1663, Concepts and Terminology for Programmers, by R. W. Engels, 2 October 1967.

6. Linebarger, R. N. and Brennan, R. D., "A Survey of Digital Simulation," Simulation, v. 3, pp. 22-36, December 1964.

7. Linebarger, R. N., "DSL/90," Simulation, v. 7, p. 108, September 1966.

8. Palevsky, M. and Howell, J. V., "The DES-1, a Real Time Digital Simulation Computer," AFIPS Conference Proceedings, 1963 Fall Joint Computer Conference, v. 25, pp. 459-472, 1963.

9. Scientific Data Systems, Manual Number 90 08 86A, DES-1 Programming Technical Manual, November 1965.

10. Scientific Data Systems, Manual Number 98 00 65A, DES-1 Reference Manual, October 1965.

11. Scientific Data Systems, Manual Number 90 00 50F, SDS 9300 Computer Reference Manual, March 1967.

12. Scientific Data Systems, "SDS Simulation Software," by Robert Gold.

13. Scientific Data Systems, Manual Number 90 05 06F, Symbol and Meta-Symbol Reference Manual, August 1967.

14. Scientific Data Systems, Program Description, model 860 791 11A00, DES-1 SYSGEN for NAA System, March 1968.

15. The SCi Simulation Software Committee, "The SCi Continuous System Simulation Language," Simulation, v. 9, pp. 281-303, December 1967.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Associate Professor G. A. Rahe, Code 52Ra Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Mr. R. L. Limes, Code 52Ec Supervisor, Electrical Engineering Computer Laboratory Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	LT Michael A. Kalashian Co B, Class 2-70 TBS, Marine Corps Base Quantico, Virginia 22134	1
6.	Commandant of the Marine Corps (Code A03C) Headquarters, U. S. Marine Corps Washington, D. C. 20380	1
7.	James Carson Breckinridge Library Marine Corps Development and Educational Command Quantico, Virginia 22134	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3. REPORT TITLE

DES-1: An Inter-active Continuous System Simulation Language

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Master's Thesis; (June 1969)

5. AUTHOR(S) (First name, middle initial, last name)

Michael Alex Kalashian

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1969	51	15

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

There are strong tutorial advantages to Digital Computer Simulation of Control System's problems. This is particularly true where such simulations do not require sophisticated programming techniques and where the user may directly interact with his problem. The purpose of this study was to develop such a capability for the Naval Postgraduate School's direct-access Computer System.

The installation was to be accomplished using the DES-1 Simulation Language and an SDS 9300 Digital Computer. The DES-1 software requires a special DES-1 Console for optimum performance. Due to the lack of this Console, a reformulation of the language was necessary. This process involved simulating the Console and revising the language to operate with existing hardware.

The language was re-written and the revised system has been installed as an operating system. Complete documentation is available in the DES-1 Programming Manual, which was prepared as part of this study.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
DES-1						
Digital Simulation						
Simulation Language						